IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A functional random instruction testing (FRIT) method for testing a ~~complex device~~ microprocessor, comprising:

converting a FRIT kernel into kernel test patterns and storing the kernel test patterns in a tester memory;

loading, at the tester, the kernel test patterns stored in the tester memory onto an on-board memory of the microprocessor to form the FRIT kernel in the on-board memory of the microprocessor, wherein the FRIT kernel includes a software built-in self-test engine (SBE); ~~a complex device under test (DUT);~~

executing by the microprocessor and from the on-board memory of the microprocessor, software within the FRIT kernel to repeatedly generate and execute functional tests to test the microprocessor; ~~on the complex device under test (DUT), to perform a re-generative functional test of the complex device under test (DUT);~~ and

comparing, at the tester, a test result of the ~~re-generative~~ functional ~~test~~ tests with an expected test result to check for manufacturing defects.

2. (Currently Amended) The method as claimed in claim 1, wherein said FRIT kernel includes ~~a software built-in self-test engine (SBE) to execute the re-generative functional test of the complex device under test (DUT), and the~~ an expected test result obtained from computer modeling of the microprocessor ~~complex device under test (DUT)~~.

3. (Canceled)

4. (Currently Amended) The method as claimed in claim 2, wherein said software built-in self-test engine (SBE) of the FRIT kernel comprises:

a ~~RIT~~ random instruction test (RIT) generator including compact RIT machine code ~~reside~~ residing in the on-board memory of the microprocessor ~~complex device under test (DUT)~~ for generating the ~~re-generated~~ functional ~~test~~ tests;

a test program execution module including test execution directives for providing an environment to store and run the ~~re-generated~~ functional ~~test~~ tests; and

a test result compaction module including compression machine code to compress test results of the ~~re-generated~~ functional ~~test~~ tests for storage in the on-board memory of the microprocessor ~~complex device under test (DUT)~~.

5. (Original) The method as claimed in claim 4, wherein said test execution environment employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops.

6. (Canceled)

7. (Currently Amended) The method as claimed in claim [[6]] 1, wherein, when software within the FRIT kernel is executed from the on-board memory, the microprocessor performs the following:

beginning a set-up for executing [[a]] the software built-in self-test engine (SBE) within the FRIT kernel;

executing the software built-in self-test engine (SBE) to generate a series of test sequences and associated data for respective test sequences;

running the test sequences, and at the end of the test sequences, obtaining the test results for storage in the on-board memory; and

dumping the test results, via an interface, for making a comparison with the expected test result to check for manufacturing defects.

8. (Currently Amended) The method as claimed in claim 7, wherein said FRIT kernel includes ~~the software built-in self-test engine (SBE) to execute the re-generative functional test of the complex device under test (DUT), and~~ the expected test result obtained either from computer modeling of the microprocessor ~~complex device under test (DUT)~~ or from a known good device.

9. (Previously Presented) The method as claimed in claim 7, wherein said software built-in self-test engine (SBE) of the FRIT kernel is programmed to generate and execute one or more ("N") instruction sequences, each sequence being executed on one or more (M) data sets, where N and M represent an integer no less than "1" and are user-specified numbers used in generating the FRIT kernel.

10. (Original) The method as claimed in claim 9, wherein said software built-in self-test engine (SBE) of the FRIT kernel is further programmed to generate one or more signatures to provide a unique identification of the test result of each test sequence and indicate whether the test result of a particular test sequence is "good" or "bad".

11. (Currently Amended) A computer readable medium having instructions stored thereon which, when executed by a system, cause the system to perform:

receiving a functional random instruction test (FRIT) kernel as kernel test patterns and storing the kernel test patterns in a tester memory;

loading the kernel test patterns stored in the tester memory onto an on-board memory of a microprocessor to form the FRIT kernel in the on-board memory of the microprocessor, wherein the FRIT kernel includes a software built-in self-test engine (SBE); ~~complex device under test (DUT);~~

enabling execution by the microprocessor and from the on-board memory of the microprocessor of software within the FRIT kernel to repeatedly generate and execute functional tests to test the microprocessor; ~~on the complex device under test (DUT), to peform a re-generative functional test of the complex device under test (DUT);~~ and

making a comparison between a test result of the ~~re-generative~~ functional ~~test~~ tests and an expected test result to check for manufacturing defects.

12. (Currently Amended) The computer readable medium as claimed in claim 11, wherein said FRIT kernel includes ~~a software built-in self-test engine (SBE) to execute the re-generative functional test of the complex device under test (DUT), and~~ the expected test result.

13. (Currently Amended) The computer readable medium as claimed in claim [[12]] 11, wherein said expected test result is obtained from computer modeling of the microprocessor ~~complex device under test (DUT)~~ or from a known good device.

14. (Currently Amended) The computer readable medium as claimed in claim [[12]] 11, wherein said software built-in self-test engine (SBE) of the FRIT kernel comprises:

a ~~RIT~~ random instruction test (RIT) generator including compact RIT machine code in the on-board memory of the microprocessor ~~complex device under test (DUT)~~ for generating the ~~re-generated~~ functional ~~test~~ tests;

a test program execution module including test execution directives for providing an environment to store and run the ~~re-generated~~ functional ~~test~~ tests; and

a test result compaction module including compression machine code to compress test results of the ~~re-generated~~ functional ~~test~~ tests for storage in the on-board memory of the microprocessor ~~complex device under test (DUT)~~.

15. (Original) The computer readable medium as claimed in claim 14, wherein said test execution environment employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops.

16. (Canceled)

17. (Currently Amended) The computer readable medium as claimed in claim [[16]] 11, wherein, when software within the FRIT kernel is executed from the on-board memory, the microprocessor performs the following:

beginning a set-up for executing [[a]] the software built-in self-test engine (SBE) within the FRIT kernel;

executing the software built-in self-test engine (SBE) to generate a series of test sequences and associated data for respective test sequences;

running the test sequences, and at the end of the test sequences, obtaining the test results for storage in the on-board memory; and

dumping the test results to the testing system, via an interface, for making said comparison with the expected test result to check for manufacturing defects.

18. (Canceled)

19. (Currently Amended) The computer readable medium as claimed in claim [[18]] 11, wherein said software built-in self-test engine (SBE) of the FRIT kernel is programmed to generate and execute one or more ("N") instruction sequences, each sequence being executed on one or more (M) data sets where N and M represent an integer no less than "1" and are user-specified numbers used in generating the FRIT kernel.

20. (Currently Amended) The computer readable medium as claimed in claim [[19]] 11, wherein said software built-in self-test engine (SBE) of the FRIT kernel is further programmed to generate one or more signatures to provide a unique identification of the test result of each test sequence and indicate whether the test result of a particular test sequence is "good" or "bad".

21. (Currently Amended) A test system for testing a ~~complex device~~ microprocessor, comprising:

a tester memory; and

a controller to test a functionality of the ~~complex device~~ microprocessor by:

receiving a functional random instruction test (FRIT) kernel as kernel test patterns and storing the kernel test patterns in the tester memory;

loading the kernel test patterns stored in the tester memory onto on-board memory of the microprocessor to form the FRIT kernel in the on-board memory of the microprocessor, wherein the FRIT kernel includes a software built-in self-test engine (SBE); ~~complex device;~~

enabling execution by the microprocessor and from the on-board memory of the microprocessor of software within the FRIT kernel to repeatedly generate and execute functional tests to test the microprocessor; ~~on the complex device, to perform a re-generative functional test of the complex device;~~ and

making a comparison between a test result of the ~~re-generative~~ functional ~~test~~ tests and an expected test result to check for manufacturing defects.

22. (Canceled)

23. (Currently Amended) The test system as claimed in claim [[22]] 21, wherein said expected test result is obtained from computer modeling of the ~~complex device~~ microprocessor or from a known good device.

24. (Currently Amended) The test system as claimed in claim [[22]] 21, wherein said software built-in self-test engine (SBE) of the FRIT kernel comprises:

a ~~RIT~~ random instruction test (RIT) generator including compact RIT machine code in the on-board memory of the microprocessor ~~complex device~~ for generating the ~~re-generated~~ functional ~~test~~ tests;

a test program execution module including test execution directives for providing an environment to store and run the ~~re-generated~~ functional ~~test~~ tests; and

a test result compaction module including compression machine code to compress test results of the ~~re-generated~~ functional ~~test~~ tests for storage in the on-board memory of the microprocessor ~~complex device~~.

25. (Original) The test system as claimed in claim 24, wherein said test execution environment employs an exception handler for handling illegal conditions such as undesirable memory accesses, deadlock, shut-down, and infinite loops.

26-38. (Canceled)